(6)

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2001-236237
(43)Date of publication of application : 31.08.2001

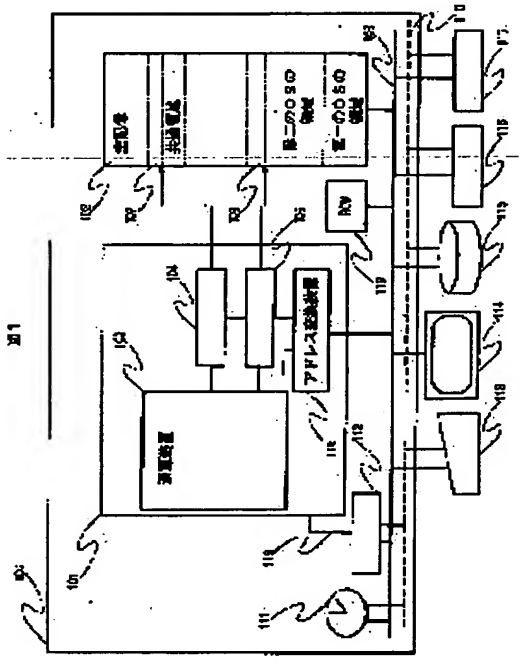
(51)Int.Cl. G06F 9/46
G06F 9/06
G06F 11/28

(21)Application number : 2000-052108
(22)Date of filing : 23.02.2000

(71)Applicant : HITACHI LTD
(72)Inventor : KIMURA SHINJI
ARAI TOSHIAKI
SATO MASAHIDE
UMETSU TOSHIKAZU

(54) METHOD FOR CONSTITUTING MULTI-OS

(57)Abstract:
PROBLEM TO BE SOLVED: To provide a method for constituting a multi-OS for reducing the generation of overhead without necessitating any special hardware.
SOLUTION: A physical memory 102 is divided for each of plural operating systems. An interruption management program independent of the operating system accepts all outside interruption, and decides which the operating system interruption handler should be started according to an interruption factor. A timing in which the interruption handler should be started is decided according to the executed state of the operating system, and the interruption handler of each operating system is started based on that.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(6)

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2001－236237
（P2001－236237A）
(43)公開日　平成13年８月31日(2001.8.31)

(51)Int.Cl.[7]		識別記号	FI			テーマコード(参考)
G06F	9/46	350	G06F	9/46	350	5B042
	9/06	410		9/06	410D	5B076
	11/28			11/28	A	5B098

審査請求　未請求　請求項の数５　ＯＬ　（全 22 頁）

(21)出願番号　特願2000－52108(P2000－52108)

(22)出願日　平成12年２月23日(2000.2.23)

(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地
(72)発明者　木村　信二
神奈川県川崎市麻生区王禅寺1099番地　株式会社日立製作所システム開発研究所内
(72)発明者　新井　利明
神奈川県川崎市麻生区王禅寺1099番地　株式会社日立製作所システム開発研究所内
(74)代理人　100075096
弁理士　作田　康夫

最終頁に続く

(54)【発明の名称】　マルチＯＳ構成方法

(57)【要約】
【課題】特殊なハードウェアを必要とせず、オーバヘッドの発生を抑えたマルチＯＳの構成方法を提供する。
【解決手段】複数のオペレーティングシステム毎に物理メモリ１０２を分割する。オペレーティングシステムから独立した割り込み管理プログラムが、すべての外部割り込みを受け付け、割り込み要因によりどのオペレーティングシステムの割り込みハンドラを起動すべきかを決定する。オペレーティングシステムの実行状態により割り込みハンドラを起動するタイミングを決定して、それに基づいて各オペレーティングシステムの割り込みハンドラを起動する

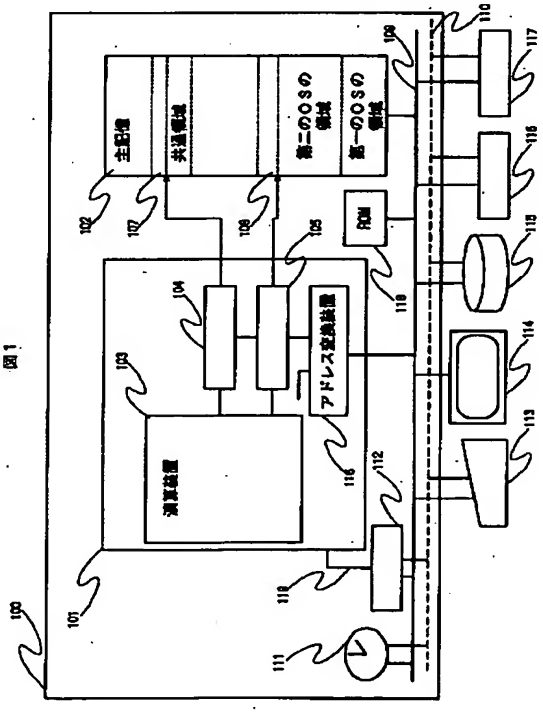


図１

【特許請求の範囲】

【請求項１】計算機システムにおけるオペレーティングシステムの制御方法において、計算機のハードウェア資源を第一のオペレーティングシステムの管理対象から外して他のオペレーティングシステムに与える手順と、複数のオペレーティングシステムで共有する領域に配置した共通の処理手順とデータを有し、前記共通処理手順の割り込み処理部が各オペレーティングシステムの実行をスケジュールすることにより、各オペレーティングシステムが実行する特権命令のエミュレート処理なしで、複数のオペレーティングシステムを一台の計算機で動作させるようにしたマルチＯＳの構成方法において、複数のオペレーティングシステムで共有する領域に配置した共通の処理手順を第一のオペレーティングシステムのデバイスドライバとして組み込む手順と、該デバイスドライバが第一のオペレーティングシステムの割り込み処理部から前記共通処理手順の割り込み処理部へ制御を切り替える手順と、該デバイスドライバが第二のオペレーティングシステムをロードして第一のオペレーティングシステムとは別の仮想アドレス空間で第二のオペレーティングシステムを起動する手順とを有し、前記共通処理手順の割り込み処理部が各オペレーティングシステムの実行をスケジュールすることにより、第一のオペレーティングシステムの処理手順を変更することなしに、複数のオペレーティングシステムを一台の計算機で動作させることを特徴とするマルチＯＳ構成方法。

【請求項２】請求項１記載のマルチＯＳ構成方法において、１つのオペレーティングシステムが使用する割り込み資源を他のオペレーティングシステムがアクセスできないようにするため、１つのオペレーティングシステムが割り込み資源を必要とする時点で、他のオペレーティングシステムに対しアクセスできないように予約することを特徴とするマルチＯＳ構成方法。

【請求項３】請求項１記載のマルチＯＳ構成方法において、１つのオペレーティングシステムに割り込み資源を割り当て後、該オペレーティングシステムが割り込み資源を必要としなくなった時点で、他のオペレーティングシステムが該割り込み資源を利用できるようにするため、他のオペレーティングシステムの割り込み資源の予約を解除することを特徴とするマルチＯＳ構成方法。

【請求項４】請求項１記載のマルチＯＳ構成方法において、複数のオペレーティングシステムで共有する領域に配置した共通処理手順及び共通データのアドレスを、すべてのオペレーティングシステムの仮想アドレス空間上で一致させる手順を有することを特徴とするマルチＯＳ構成方法。

【請求項５】請求項１記載のマルチＯＳ構成方法において、被監視オペレーティングシステムの関数群のアドレス情報を取得する手順と、アドレス情報からオペレーティングシステム内で障害発生時にコールされるエラー関数のアドレスを抽出する手順と、抽出したエラー関数のアドレスに割り込み発生命令を書きこむ手順と、該割り込みが発生したとき、複数のオペレーティングシステムで共有する領域に配置した共通処理手順の割り込み処理部が監視オペレーティングシステムに通知する手順と、通知によって該監視オペレーティングシステムが被監視オペレーティングシステムの障害診断あるいは障害回復処理を行う手順を有することを特徴とするマルチＯＳ構成方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、一台の計算機上で複数のオペレーティングシステムを稼動させるためのマルチＯＳ構成方法に関する。

【０００２】

【従来の技術】通常、計算機システムでは、計算機上で１つのオペレーティングシステムを動作させ、それにより計算機のプロセッサ、メモリ、および、二次記憶装置等の計算機資源を管理し、計算機が効率良く動作できるように資源スケジュールを実施している。オペレーティングシステムには様々な種類がある。バッチ処理に優れるものや、ＴＳＳ(Time Sharing System)に優れるもの、ＧＵＩ(Graphical UserInterface)に優れているものなど様々である。

【０００３】一方、これら複数あるオペレーティングシステムを一台の計算機で同時に実行したいというニーズがある。例えば、大型計算機においては、実際の業務に伴うオンライン処理を実行するオペレーティングシステムと、開発用のオペレーティングシステムを一台の計算機で動作させたいという要求がある。あるいは、ＧＵＩの整っているオペレーティングシステムと、実時間性に優れているオペレーティングシステムを同時に稼動させたいという要求もある。

【０００４】しかしながら、個々のオペレーティングシステムは、単独で計算機資源の管理を実施することを仮定しており、複数のオペレーティングシステムの共存は、何らかの機構なしに実現することは困難である。

【０００５】一台の計算機上で複数のオペレーティングシステムを動作させる機構として、大型計算機で実現されている仮想計算機方式がある。仮想計算機方式では、仮想計算機制御プログラムが全ハードウェア資源を占有して管理し、それを仮想化して仮想計算機を構成する。仮想計算機を構成する制御部は、物理メモリ、入出力機器装置、外部割り込み等を仮想化する。

【０００６】例えば、分割された物理メモリは、仮想計算機に対してはあたかも０番地から始まる物理メモリのように振る舞い、入出力装置を識別する装置番号も同様に仮想化されている。更に、仮想計算機方式では、磁気ディスクの記憶領域も分割して磁気ディスク装置の仮想化まで実現している。

【０００７】各オペレーティングシステムは、制御プログラムにより構築された仮想計算機上で実行されるように制御プログラムによりスケジュールされる。

【０００８】

【発明が解決しようとする課題】上述したような従来の大型計算機における仮想計算機方式では、計算機資源を完全に仮想化、および、分割しようとするため、仮想計算機を構成する制御部分が複雑になるという問題がある。

【０００９】また、特別なハードウェア支援がない場合、仮想計算機上で動作するオペレーティングシステムが発行する制御レジスタの設定、入出力命令等の特権命令は、仮想計算機制御プログラムによりエミュレートしなければならない。このため、オーバーヘッドが大きくなるという問題もある。実際、仮想計算機を実装している大型計算機では、仮想計算機用に特別なプロセッサ機能やマイクロコード等のハードウェアを追加してオーバーヘッドの削減を図っている。

【００１０】このように、仮想計算機方式は、完全に計算機資源を仮想化することを目的としているため複雑であり、更に、仮想計算機の高性能化のためには特殊なハードウェア機構が必要となる。

【００１１】一方、一台の計算機上で複数のオペレーティングシステムのインタフェースを提供する技術として、マイクロカーネルと呼ばれる技術がある。マイクロカーネルでは、マイクロカーネルの上に、ユーザに見せるオペレーティングシステム機能を提供するオペレーティングシステムサーバが構築される。ユーザは、そのサーバを経由して計算機資源を利用する。オペレーティングシステム毎のサーバを用意すれば、ユーザに様々なオペレーティングシステム環境を提供できる。

【００１２】しかし、マイクロカーネル方式では、オペレーティングシステムサーバをマイクロカーネルに合わせて新規に構築する必要がある。多くの場合、現在あるオペレーティングシステムをマイクロカーネル上で動作するように変更することになるが、スケジューリング、メモリ管理等のカーネルの中枢部分も変更することになり、変更箇所が多く、また、変更箇所がオペレーティングシステムの中枢部分に及ぶため変更作業が複雑で容易ではないといった問題がある。

【００１３】また、オペレーティングシステムサーバはマイクロカーネルのサービスを利用することになるが、これは通常のオペレーティングシステムではないことであり、オーバーヘッドとなり性能低下をもたらす。

【００１４】本発明の目的は、このような従来技術における問題点を解決し、特殊なハードウェアを用いることなく、１台の計算機上で複数のオペレーティングシステムの同時・並列的な実行を実現することにある。

【００１５】また、本発明の他の目的は、１台の計算機上で複数のオペレーティングシステムを動作させるにあたり、特権命令のエミュレーションによるオーバヘッドの発生を抑えることにある。

【００１６】

【課題を解決するための手段】上述した目的を達成するために、本発明によるマルチＯＳの構成方法では、複数のオペレーティングシステム毎に物理メモリを分割する。オペレーティングシステムから独立した割り込み管理プログラムが、すべての外部割り込みを受け付け、割り込み要因によりどのオペレーティングシステムの割り込みハンドラを起動すべきかを決定する。オペレーティングシステムの実行状態により割り込みハンドラを起動するタイミングを決定して、それに基づいて各オペレーティングシステムの割り込みハンドラを起動することにより、複数のオペレーティングシステムを一台の計算機で動作させる。

【００１７】本発明によれば、第一のオペレーティングシステムを補完する機能を容易に追加でき、高機能な計算機システムの構築が可能になる。更に、デバイスドライバとは異なり、第一のオペレーティングシステムとは全く独立して動作する機能を組み込むことができるため、第一のオペレーティングシステムに依存しない高信頼化機能を追加することも可能になる。

【００１８】また、マイクロカーネル方式で複数のマルチオペレーティングシステム環境を構築する方法では、それぞれのオペレーティングシステムのインタフェースを提供するオペレーティングシステムサーバの構築が難しいという問題がある。本発明によれば、オペレーティングシステムを変更することなく、割り込み管理部分の追加だけで、簡単にマルチオペレーティングシステム環境を構成できる。

【００１９】

【発明の実施の形態】図１は本発明の実施の形態における計算機１００の構成を示すブロック図である。

【００２０】計算機１００は、プロセッサ１０１、主記憶装置１０２、バス１０９、割り込み信号線１１０、クロック割り込み生成器１１１、割り込み制御装置１１２、ブート手順を格納している記憶装置１１８、および、割り込みバス１１９を有して構成されている。

【００２１】割り込み信号線１１０は、外部の入出力機器と割り込み制御装置１１２を接続している。外部機器が割り込みを発生すると、割り込み信号線１１０を経由して割り込み制御装置１１２が信号を受け取る。割り込み制御装置１１２は、この信号を数値化して、割り込みバス１１９を介してプロセッサ１０１に渡す。

【００２２】クロック割り込み生成器１１１は、周期的な割り込みを生成する。

【００２３】割り込み制御装置１１２は、外部機器からの割り込みを受け付け、要求元にしたがって数値化された割り込み信号を生成し、プロセッサ１０１に送る。また、プロセッサ１０１からの指示により、特定の機器か

らの割り込み信号をプロセッサ１０１に通知しないようにすることもできる。

【００２４】プロセッサ１０１は、演算装置１０３、割り込みテーブルレジスタ１０４、ページテーブルレジスタ１０５、および、アドレス変換装置１０６を有して構成されている。

【００２５】割り込みテーブルレジスタ１０４は、割り込みテーブル１０７の仮想アドレスを指し示している。割り込みテーブルの詳細については後述するが、割り込み番号毎の割り込みハンドラの開始アドレスを記録している。図１で、割り込みレジスタ１０４と割り込みテーブル１０７の接続を破線で記載しているのは、割り込みテーブルレジスタ１０４が割り込みテーブルの指し示すためである。

【００２６】割り込みが発生すると、プロセッサ１０１は、割り込み制御装置１１２から数値化された割り込み番号を受ける。プロセッサ１０１は、この番号をインデックスとして割り込みテーブル１０７より割り込みハンドラアドレスを取得し、割り込みハンドラに制御を渡す。

【００２７】ページテーブルレジスタ１０５は、ページテーブル１０８を指し示している。ページテーブルレジスタ１０５は、ページテーブル１０８の物理アドレスを格納している。

【００２８】アドレス変換装置１０６は、演算装置が要求する命令アドレス、あるいは、オペランドが格納されているアドレスを受け取り、ページテーブルレジスタ１０５の指しているページテーブル１０８の内容に基づき仮想－実アドレス変換を実施する。

【００２９】計算機１００には、外部入出力装置としてキーボード１１３、ディスプレイ１１４、磁気ディスク１１５、その他の外部機器１１６、および、１１７が接続されている。ディスプレイ１１４を除く機器は割り込み信号線１１０により割り込み制御装置１１２に接続されている。

【００３０】主記憶装置１０２の内容について簡単に説明する。ここでは、計算機１０１では２つのオペレーティングシステムが動作しているとする。それぞれを第一のオペレーティングシステム、第二のオペレーティングシステムと呼ぶことにする。また、計算機を起動すると第一のオペレーティングシステムが起動するように設定されており、外部機器１１６、および、１１７は第二のオペレーティングシステムにより管理される機器とする。以下では、第一のオペレーティングシステムを第一のＯＳ、第二のオペレーティングシステムを第二のＯＳと記す。

【００３１】第一のＯＳは、起動時の初期化処理で、その他のオペレーティングシステム用に、この場合は第二のＯＳ用に物理メモリ領域を予約する。つまり、第一のＯＳが、第二のＯＳ用に予約した物理メモリ領域を利用できないように物理メモリ領域を確保する。図１は、この奪った領域に第二のＯＳがロードされている様子を示している。

【００３２】また、主記憶装置１０２は、すべてのオペレーティングシステムから参照可能な共通領域を持つ。共通領域には、割り込みテーブル１０７、割り込み管理プログラム、割り込みハンドラ、各オペレーティングシステムから呼び出し可能なインタフェースモジュール等が格納される。

【００３３】本実施形態の動作概要について説明する。本実施形態では、第二のＯＳは第一のＯＳよりも優先して動作する。優先して動作するとは、第一のＯＳは第二のＯＳがアイドル状態であるときのみ動作可能であることを示す。第二のＯＳの処理が終了しない限り、第一のＯＳは動作できない。

【００３４】また、第二のＯＳが管理するデバイスが割り込みを発生すると、第一のＯＳの処理は中断され、制御は第二のＯＳに移る。第二のＯＳ実行中に第一のＯＳへの割り込みが発生しても、その割り込み処理は第一のＯＳが実行されるまで延期される。

【００３５】さらに、第一のＯＳと第二のＯＳは明確に区分されており、割り込みハンドラなどを配置する共通領域以外は、互いにアクセスできないように構成される。これにより、２つのオペレーティングシステムが誤って互いの領域にアクセスして障害が起きることを防いでいる。

【００３６】図２は、本実施形態における２つのオペレーティングシステムの関係を概念的に示した模式図である。

【００３７】それぞれのオペレーティングシステムは、それぞれ独立したアドレス空間を保持する。２０１は第一のＯＳの仮想空間で、２０２は第二のＯＳの仮想空間を示している。ここで、第二のＯＳの空間２０２に対応する実記憶は、図１の主記憶装置１０２の第二のＯＳの領域になる。

【００３８】仮想空間の一部分には、共通領域２０３がマップされる。共通領域２０３に対応する実記憶は、図１において、主記憶１０２の共通領域として示される領域である。

【００３９】図２は、また、各オペレーティングシステムが管理するハードウェアを示している。第一のＯＳは、キーボード１１３、ディスプレイ１１４、および、磁気ディスク１１５を、第二のＯＳは入出力装置１１６、および、１１７を管理する。クロック１１１と割り込み制御装置１１２は、もともとは第一のＯＳが管理しているハードウェアであるが、共通領域２０３中のプログラムが管理する。

【００４０】図３は、本実施形態でのページテーブルの構成を示している。

【００４１】ページテーブル３００は、プロセッサ１０

１の仮想アドレス空間の仮想ページ毎に、それぞれの仮想ページを記述するエントリを持っている。それぞれのエントリは、有効ビット３０１と、物理ページ番号３０２を含んで構成される。

【００４２】有効ビット３０１は、その仮想ページに対応する物理ページが割り当てられているか、つまり、仮想－実アドレス変換が可能かを示している。例えば、ページテーブル３００の仮想ページ３は、有効ビットがセットされていないので、仮想ページ３に対応する物理ページが存在しないことを示している。有効ビット３０１がセットされていない仮想ページへのアクセスが発生すると、プロセッサはページフォルトを発生する。

【００４３】物理メモリ番号３０２は、仮想ページに対応する物理ページ番号を記録している。

【００４４】アドレス変換装置１０６は、ページテーブルレジスタ１０５の指し示しているページテーブルの内容を参照して、演算装置１０３の生成する仮想アドレスを実アドレスに変換する。プロセッサ１０１は、変換により得られた実アドレスにより主記憶装置１０２を参照する。

【００４５】ページテーブルを切り替えることにより独立した空間を構築することができ、図２に示した第一のＯＳの空間、および、第二のＯＳの空間の構築が可能である。また、共通領域２０３については、両方のオペレーティングシステムのページテーブルの共通領域に対応する部分に、同じ物理ページをマップするように設定しておけば、共通領域を実現できる。

【００４６】図４は、割り込みテーブルの構成を示している。

【００４７】割り込みテーブル４００は、プロセッサ１０１が割り込み制御装置１１２から受ける割り込み番号毎の、割り込みハンドラの仮想アドレス４０１を記録している。プロセッサ１０１は、割り込み要求を割り込み制御装置１１２から受け取ると、割り込み番号に対応する割り込みハンドラのアドレスを、割り込みテーブルレジスタ１０４の指し示している割り込みテーブル４００から取得し、そのアドレスに制御を移すことで割り込み処理を開始する。

【００４８】図５は、割り込み制御装置１１２の構成を示している。

【００４９】割り込み制御装置１１２は、割り込みマスクレジスタ５０１、および、選択装置５０２を持っている。

【００５０】割り込みを発生する入出力装置は、割り込み信号線１１０により割り込み制御装置１１２と接続される。入出力機器の発生する割り込みは、割り込み信号線１１０のどの信号線に接続するかにより優先順位が付けられる。ここでは、割り込み０番に対応する割り込み信号がもっとも優先度が高い割り込みであるとする。

【００５１】割り込み信号線１１０は、選択装置５０２に接続されている。選択装置５０２は、割り込み信号を受けると、プロセッサがその割り込みを受けたことを通知するまで、未処理の割り込みがあることを記録している。

【００５２】割り込みマスクレジスタ５０１は、入出力機器の発生した割り込みをプロセッサに通知してよいかを記録している。割り込みマスクレジスタ５０１は、プロセッサ１０１からの入出力命令により設定可能である。

【００５３】選択装置５０２は、割り込み信号線１１０からの割り込み要求を受けた時と、割り込みマスクレジスタ５０１の内容が書き換えられた時に、選択装置５０２が記録している未処理割り込みと、割り込みマスクレジスタ５０２の内容を比較して、プロセッサに割り込みを通知するかどうかを決める。具体的には、選択装置５０２が記録している未処理割り込みのうち、割り込みマスクレジスタ５０１に割り込み可能と設定されていて、優先度の最も高い割り込みから順にプロセッサに通知する。選択した割り込みについて、選択装置５０２は、通知する割り込み信号に対応する数値信号を、割り込みバス１１９経由でプロセッサ１０１に送る。

【００５４】プロセッサ１０１は、割り込みを受けたときに、入出力命令により選択装置５０２に記録されている未処理割り込みを解消できる。

【００５５】次に、本実施形態における、計算機のブート手順について説明する。

【００５６】ブート手順の始めの部分は、読み取り専用の記憶装置１１８に格納されている。記憶装置１１８は、プロセッサの物理アドレス空間のある決められたアドレスにマップされるように、バス１０９を介してプロセッサ１０１に接続されている。この手順は、ハードウェア構成の検出、オペレーティングシステムをロードするプログラムの主記憶へのローディングを実施する。

【００５７】プロセッサ１０１がリセットされると、プロセッサ１０１は予め定められた物理アドレスに制御を移す。記憶装置１１８は、この時に実行されるプログラムを格納しており、プロセッサ１０１がリセットされた時にこのプログラムに制御を渡せるように物理アドレス空間にマップされている。

【００５８】記憶装置１１８に格納されているプログラムは、磁気ディスク装置１１２に格納されている第一のＯＳのカーネルローダを主記憶装置１０２にロードして実行する。カーネルローダは、磁気ディスク装置１１２の予め定められた位置にあり、記憶装置１１８に格納されていたプログラムは、容易にこれを見つけることができる。

【００５９】図６は、本実施形態における、オペレーティングシステムのカーネルローダの処理手順を示すフローチャートである。

【００６０】カーネルローダは、オペレーティングシス

テムのファイルシステム構造を理解して、ファイル名によりファイルの格納位置を特定し、主記憶に読み込むことができるように構成されている。

【００６１】カーネルローダは、まず、カーネルにパラメータとして渡す主記憶リスト８０１、ロードモジュールリスト８０４、および、デバイスリスト８０２を初期化し、カーネル用のページテーブル領域を割り当てる（ステップ６０１）。

【００６２】主記憶リスト８０１は、主記憶装置１０２の利用状況を示すデータ構造であり、カーネルローダが以降の処理で物理メモリの割り当てをする場合は、主記憶リスト８０１を参照、及び、変更して実施する。

【００６３】次に、カーネル用のハードウェア構成の検査（ステップ６０２）、および、ハードウェア構成データの作成（ステップ６０３）を実施する。ステップ６０２においては、計算機１００にどのようなハードウェアデバイスが接続されているかをハードウェアが検査する。続くステップ６０３では、ステップ６０２の結果に基づいてハードウェア構成に関するデータ構造であるデバイスリスト８０２が作成される。オペレーティングシステムのカーネルは、このデバイスリスト８０２を参照してカーネル初期化を実施する。

【００６４】次に、オペレーティングシステムの構成情報７００を磁気ディスク装置１１２より読み込み、パラメータテーブル８１０に構成情報のアドレスを設定する（ステップ６０４）。オペレーティングシステムのカーネルは、カーネル本体のファイルと、その他のデバイスドライバのファイルといったように、複数のファイルから構成されていてもよい。構成情報７００は、予め決められたファイル名で磁気ディスク装置１１２に格納されており、ロードプログラムはこれを見つけることができる。

【００６５】本実施形態におけるカーネル構成情報のデータ構造を図７に示す。

【００６６】カーネル構成情報７００は、カーネルローダやオペレーティングシステムが参照するデータを格納している。格納されているデータには名前が付けられており、プログラムは名前からそれに対応するデータを取得することができる。図７の例では、名前がメモリサイズというエントリ７０１があり、その内容としてメモリサイズを示すデータが格納されている。また、名前がsecondary OSというエントリ７０５は、第二のＯＳ用のデータが格納されている。

【００６７】カーネル構成情報７００を読み込んだ後、カーネルローダは、カーネル構成情報７００中のメモリサイズ７０１に格納されたデータの値によって、第一のＯＳに割り当てる物理メモリサイズを決定する（ステップ６０５）。次に、オブジェクトファイル７０３に設定されているカーネル構成ファイルのすべてを主記憶装置１０２に読み込み、ロードモジュールリスト８０４にエントリを追加する（ステップ６０６～６０８）。ここではｋｅｒｎｅｌ、ｄｒｉｖｅｒ１、および、ｄｒｉｖｅｒ２というファイル名のオブジェクトファイルをロードする。

【００６８】次にステップ６０５で求めたメモリサイズに対応するカーネル用のページテーブルを設定する（ステップ６０９）。ステップ６０９で設定したページテーブルのアドレスをページテーブルレジスタ１０５に設定し、プロセッサを仮想アドレスモードに移行させ（ステップ６１０）、構築した主記憶リスト８０１、デバイスリスト８０２、カーネル構成情報テーブル８０３、および、ロードオブジェクトリスト８０４の組からなるパラメータテーブル８１０をパラメータとして、カーネルの初期化ルーチンに制御を渡す（ステップ６１１）。カーネルのエントリポイントはカーネルファイル内のデータに記録されている。

【００６９】次に、ステップ６０１から始まるブート手順が作成するハードウェア構成データと、ロードオブジェクトデータの構成について説明する。図８は、ハードウェア構成データとロードオブジェクトデータの構成を示す図である。

【００７０】パラメータテーブル８００は、カーネルローダが作成するデータ構造である。パラメータテーブル８００から始まる３つのリストは、ローダが構築するカーネルの仮想空間に配置されており、カーネルから参照可能である。

【００７１】パラメータテーブル８００は、ローダが構築した３つのリストの先頭へのポインタと、１つのテーブルへのポインタを保持している。３つのリストとは、主記憶リスト８０１、デバイスリスト８０２、および、ロードオブジェクトリスト８０４で、１つのテーブルはカーネル構成情報テーブル８０３である。それぞれについて説明する。

【００７２】主記憶リスト８０１は、主記憶ブロック記述データ８１０のリストである。主記憶ブロック記述データ８１０は、ベースアドレス８１１、ブロックサイズ８１２、ブロック利用状況８１３、および、次の主記憶ブロック記述データへのポインタ８１４から構成されている。

【００７３】主記憶ブロック記述データ８１０は、ある連続した主記憶領域についての利用状況を記録している。ベースアドレス８１１は連続領域の開始物理アドレスを示し、ブロックサイズ８１２は連続領域の大きさを格納し、ブロック利用状況８１３は、当該連続領域が未使用であるか、あるいは、ローダにより割り当て済みであるかを示す値が格納されている。そして、次エントリへのポインタ８１４によりリストを構成している。図８では、８１０の次のエントリは８２０である。主記憶リスト８０１を参照することで、物理メモリの利用状態を知ることができる。

【００７４】デバイスリスト８０２は、カーネルローダが検出したハードウェアデバイスに関するデータを格納しており、ステップ６０３で作成される。デバイスリスト８０２は、デバイスデータ８５０からなるリストである。デバイスデータ８５０は、デバイスタイプ８５１、デバイス情報８５２、および、次のデバイスデータへのポインタ８５３から構成される。

【００７５】デバイスタイプ８５１には、このデバイスデータエントリ８５０により記述されるデバイスの種類を示す値が格納されている。デバイス情報８５２は、デバイスの種類に特有なデータを格納している。例えば、割り込み番号やＩ／Ｏアドレスなどがそれに相当する。そして、次のエントリへのポインタ８５３によりリストを構成している。

【００７６】カーネル構成情報テーブルへのポインタ８０３は、カーネルローダが主記憶装置１０２に読み込んだカーネル構成情報ファイル７００の内容を指し示している。

【００７７】ロードオブジェクトリスト８０４は、カーネルローダが主記憶にロードしたオブジェクトファイルに関するデータを保持している。ロードオブジェクトリストは、ロードオブジェクトデータ８３０のリストである。ロードオブジェクトデータ８３０は、オブジェクトファイル名８３１、オブジェクトアドレス８３２、および、次のロードオブジェクトデータへのポインタ８３３から構成される。

【００７８】オブジェクトファイル名８３１は、ロードオブジェクト８３０により記述されているオブジェクトファイルのファイル名である。オブジェクトアドレス８３２は、当該オブジェクトファイルのヘッダ領域がロードされているカーネル空間のアドレスを格納している。そして、次エントリへのポインタ８３３によりリストを構成している。

【００７９】ロードオブジェクトリスト８０４は、カーネルローダがカーネルを構成するオブジェクトファイルを読み込むときに同時に作成される（ステップ６０８）。

【００８０】次に、本実施形態におけるデバイスドライバのロード処理手順について説明する。デバイスドライバとは、計算機１００に接続されたハードウェアデバイスを制御するためのソフトウェアであり、図６で説明したオペレーティングシステムのカーネルの初期化中、あるいは、オペレーティングシステムが立ち上がった後にロードできるものとする。

【００８１】まず、デバイス管理テーブルについて説明する。図９は、第一のＯＳのデバイス管理テーブルの構造を示した図である。デバイス管理テーブルは、割り込み管理テーブル９００と、Ｉ／Ｏアドレス管理リスト９１０の２つのデータ構造からなる。

【００８２】割り込み管理テーブル９００は、プロセッサ１０１が受け付ける各割り込み番号について、第一のＯＳがその割り込み番号を利用するかどうかを示す値が格納されている。カーネルは、デバイスドライバがロード時に割り込み番号を要求した場合に、このテーブル９００を検査し、使用されていない場合のみ、要求された割り込み番号を使用する権利をデバイスドライバに与える。既に利用済みであると記されている場合は、そのハードウェアデバイスは第一のＯＳからは利用できないことになる。例えば、割り込み番号３の場合、番号３に対応するデータ９０２が割り当て済みかあるいは未使用かを示す値を保持する。

【００８３】Ｉ/Ｏアドレス管理リスト９１０についても同様である。Ｉ／Ｏアドレス管理リスト９１０は、Ｉ／Ｏアドレスを表現するエントリ９２０からなるリストである。エントリ９２０は、第一のＯＳが利用するＩ／Ｏアドレス９２１と、リストを構成するための次のエントリへのポインタ９２２からなる。割り込みベクタ管理テーブル９００と同様、デバイスドライバがＩ／Ｏアドレス範囲を要求した場合、カーネルは、そのアドレス範囲が既に利用されているかＩ／Ｏアドレス管理リスト９１０により検査し、未使用である場合、このリスト９１０にエントリを追加して、利用許可を与える。

【００８４】図１０は、デバイスドライバのロード処理手順を示すフローチャートである。

【００８５】デバイスドライバのロード処理では、まず、デバイスドライバの情報が取得される（ステップ１００１）。デバイスドライバの情報とは、デバイスドライバのオブジェクトファイル名であり、デバイスドライバがオペレーティングシステムのカーネル初期化時にロードされる場合は、カーネル構成情報ファイル７００のエントリ名オブジェクトファイル７０３から得られ、また、カーネル立ち上げ後にデバイスドライバがロードされる場合は、オペレータがロードを指示したオブジェクトファイル名が使用される。

【００８６】次に、指定されたオブジェクトファイルを調べ、当該オブジェクトをロードするために必要なメモリサイズを調べる（ステップ１００２）。調べたメモリサイズに必要な主記憶装置１０２上のメモリをオペレーティングシステムに対して要求し（ステップ１００３）、当該デバイスドライバをメモリに読み込む（ステップ１００４）。

【００８７】次に、当該デバイスドライバのオブジェクトファイルからデバイスドライバの初期化処理のエントリポイントを得て、デバイスドライバの初期化処理を実行する（ステップ１００５）。ステップ１００５の初期化処理では、デバイスリスト８０２を参照することにより、当該デバイスドライバが対象とするハードウェアデバイスが計算機１００に接続されているか確認する（ステップ１００６）。

【００８８】当該ハードウェアデバイスが接続されてい

るならば、当該ハードウェアデバイスの初期化、および、デバイスドライバの処理に必要なテーブルの初期化を実行する（ステップ１００７）。もし、当該ハードウェアデバイスが割り込み番号あるいはＩ／Ｏアドレスを必要とするならば、当該デバイスドライバからオペレーティングシステムに対して、割り込み番号あるいはＩ／Ｏアドレス範囲を使用することを通知する。通知を受けたオペレーティングシステムは、図９の割り込み管理テーブル９００とＩ／Ｏアドレス管理リスト９１０を調べ、要求された資源が利用可能な場合、デバイスドライバに使用の許可を与え、また、デバイスドライバ管理テーブルを更新することによって、他のデバイスドライバはこの資源を使用しないように設定する（ステップ１００８）。割り込み番号の使用の許可が得られた場合、割り込みを処理するための割り込みハンドラ処理のアドレスをオペレーティングシステムに対して登録する（ステップ１００９）。また、必要に応じて、当該デバイスドライバに固有の処理を実行する（ステップ１０１０）。

【００８９】デバイスドライバのロード処理では、ステップ１００１から１００９の処理がすべて正しく行われた場合、ロードオブジェクトリスト８０４に当該デバイスドライバのオブジェクトファイル情報を追加し（ステップ１０１１）、当該デバイスドライバがオペレーティングシステムの一部となって動作し、ハードウェアデバイスを制御する。

【００９０】図１１は、デバイスドライバのアンロード処理のフローチャートである。

【００９１】デバイスドライバのアンロードは、ハードウェアデバイスの使用を止めるために行うもので、オペレーティングシステム立ち上げ後、ロードされたデバイスドライバを、指示によりアンロードする場合や、ハードウェアデバイスが計算機から取り除かれた場合に実行される。

【００９２】デバイスドライバのアンロード処理では、まず、オペレーティングシステムにより当該デバイスドライバの終了処理が呼び出される（ステップ１１０１）。

【００９３】デバイスドライバの終了処理では、割り込みハンドラの登録、および、使用していた割り込み番号あるいはＩ／Ｏアドレス範囲の使用が解除される（ステップ１１０２、１１０３）。これにより、解除された割り込み番号あるいはＩ／Ｏアドレス範囲は未使用となり、他のデバイスドライバでの利用が可能になる。次に当該デバイスドライバが使用していたメモリを解放し（ステップ１１０４）、デバイスドライバの終了処理が完了する。

【００９４】デバイスドライバの終了処理後、オペレーティングシステムは、デバイスドライバのオブジェクトファイルをロードしていた主記憶装置１０２上のメモリを開放し（ステップ１１０５）、ロードオブジェクトリスト８０４から当該デバイスドライバのオブジェクトファイルのリストを削除する（ステップ１１０６）。

【００９５】次に、１台のＰＣ上で複数のオペレーティングシステムを実行するための処理手順について説明する。本実施形態では、マルチＯＳを実行するための制御を第一のＯＳのデバイスドライバ（以下ではマルチＯＳドライバと記す）として組み込むものとする。

【００９６】図１２にマルチＯＳドライバの処理構造を示す。

【００９７】マルチＯＳドライバの構造は大きく２つに分かれ、オペレーティングシステムから呼び出しによって実行される入り口関数部分１２０１と、入り口関数部分１２０１からの呼び出しによって実行される実行関数部分１２０２で構成される。

【００９８】マルチＯＳドライバのすべての入り口関数は対応する実行関数のアドレスを保持しており、オペレーティングシステムから入り口関数が呼び出されると、対応する実行部関数を呼び出し（ステップ１２０３）、呼び出された実行関数が当該関数の実際の処理を実行する（ステップ１２０４）。マルチＯＳドライバがこのように２つに分かれた構造になっている理由は、オペレーティングシステムがデバイスドライバをロード後、さらに、実行関数部分１２０２のアドレスを変更可能とするためである。

【００９９】図１３にマルチＯＳドライバのロード処理手順のフローチャートを示す。マルチＯＳドライバもデバイスドライバの一種であり、オペレーティングシステムは図１０で説明したフローチャートに従ってマルチＯＳドライバのオブジェクトファイルをロードする。このため、ここでは、マルチＯＳドライバのデバイスドライバ固有処理（ステップ１０１０）の詳細のみを示している。

【０１００】図１３において、まず、すべてのオペレーティングシステムから参照可能な共通領域用の物理メモリを確保する（ステップ１３０１）。この共通領域には、マルチＯＳドライバの一部として、割り込みテーブル１０７、割り込み管理プログラム、割り込みハンドラ、各オペレーティングシステムから呼びだし可能なインターフェイスモジュール等が格納される。共通領域用の物理メモリの確保は、主記憶リスト８０１を参照し、空き領域を確保する。本実施形態では、物理メモリは第一のＯＳがロードされているだけであり、空いている物理メモリのうち、アドレスの大きいほうのメモリを必要な容量だけ確保する。

【０１０１】次に、共通領域のための、第一のＯＳの仮想アドレス空間を確保する（ステップ１３０２）。この仮想アドレス空間はすべてのオペレーティングシステムで一致させる必要があるため、予めすべてのオペレーティングシステムで取り決めたアドレスを割り当てるものとする。本実施形態では、第１のＯＳの仮想アドレス空

間の中で最もアドレスの大きい空間から必要な容量だけ確保する。

【０１０２】物理メモリと仮想アドレス空間の領域が確保できたならば、ステップ１３０２で確保した仮想アドレスからステップ１３０１で確保した物理メモリにアクセスが可能なように、第一のＯＳのページテーブルを設定する（ステップ１３０３）。

【０１０３】次に、マルチＯＳドライバが第一のＯＳがロードされた領域のオブジェクトとデータを、共通領域に複写し（ステップ１３０４）、図１２で説明したマルチＯＳドライバ関数のうちのオペレーティングシステムから呼び出される入り口関数が保持する実行関数のアドレスを共通領域に複写した実行関数のアドレスに更新する（ステップ１３０５）。この実行関数のアドレスの変更により、第一のＯＳからマルチＯＳドライバの関数を呼び出した場合、共通領域にある実行関数のオブジェクトが実行されることになる。

【０１０４】図１４に、仮想アドレス空間と主記憶装置１０２の物理メモリの対応を示す。

【０１０５】１４０１は物理メモリを、１４０２は第一のＯＳの仮想アドレス空間を示しており、図中、下部から上部に向かってアドレスが大きくなっていくものとする。第一のＯＳの仮想アドレス空間１４０２は、オペレーティングシステムのみがアクセス可能なカーネル空間１４０４とオペレーティングシステムとユーザプログラムがアクセス可能なユーザ空間に分かれる。カーネル空間１４０４の下位のアドレス側１４０８は、物理メモリ１４０１のうち、第一のＯＳに割り当てたメモリ１４０９にアクセスできるようにページマップテーブルが設定されていることを示している。

【０１０６】マルチＯＳドライバは、第一のＯＳによって、物理メモリ１４０９の一部の領域１４１３にロードされ、仮想アドレス空間１４０４のマルチＯＳドライバＡ領域１４１２にマップされる。マルチＯＳドライバＡ領域１４１２は、オペレーティングシステムの仮想アドレス空間の利用状況によって変化するため、予め予測することはできない。

【０１０７】マルチＯＳドライバは、すべてのオペレーティングシステムで使用するオブジェクトとデータを含むため、すべてのオペレーティングシステムで確保でき、かつ、一致した仮想アドレス空間の領域（マルチＯＳドライバＢ領域１４１５）にオブジェクトとデータを移動させる。これにより、すべてのオペレーティングシステムで共通のアドレスでマルチＯＳドライバにアクセスできることになる。また、マルチＯＳドライバＢ領域１４１５の物理メモリも、第一のＯＳに割り当てた物理メモリ内の領域１４１３から共通領域２０３に移動させることによって、オペレーティングシステムの不具合による誤ったメモリアクセスを防ぐことができる。

【０１０８】図１５は、第二のＯＳのロード手順を示すフローチャートである。

【０１０９】この処理では、まず、第二のＯＳ用に割り当てた物理メモリ領域に、第二のＯＳのオブジェクトファイルを読み込む必要がある。しかし、第二のＯＳの物理メモリ領域は、そのままでは第一のＯＳから書き込むことはできないので、割り当てた物理メモリ領域を第一のＯＳの仮想空間に一時的にマッピングする（ステップ１５０１）。

【０１１０】ステップ１５０２では、マッピングした領域に、第一のＯＳのファイル読み込み手順を利用して、第二のＯＳのオブジェクトファイルを読み込む。

【０１１１】次に、第二のＯＳ用のページテーブルを作成する（ステップ１５０３）。このページテーブルも第二のＯＳ用の領域に作成する。このとき共通領域２０３の領域についても、第二のＯＳの空間から参照できるように、ページテーブルを構築する。

【０１１２】共通領域２０３をマッピングするマルチＯＳドライバＣ領域１４１６は、第一のＯＳのマルチＯＳドライバＢ領域１４１５の仮想アドレスと同じ値になるように設定する。

【０１１３】次に、第二のＯＳからマルチＯＳドライバＣ領域１４１６に配置したマルチＯＳドライバのインタフェースモジュールの呼び出しができるように、インタフェースモジュールのアドレスを第二のＯＳの領域に書き込み、起動時の引数として第二のＯＳに渡す（ステップ１５０４）。同様に、マルチＯＳドライバから第二のＯＳを呼び出すときに必要な関数のアドレスを、オブジェクトファイルから取得し、共通領域２０３に保持しておく（ステップ１５０５）。

【０１１４】これで、第二のＯＳ領域の設定を終了し、第一のＯＳにマッピングした、第二のＯＳ用の物理メモリ領域のマッピングを解除する（ステップ１５０６）。

【０１１５】次に、ＯＳコンテクストテーブル１６１０の第二のＯＳのコンテクストと、ＯＳ識別変数１６３０を設定する（ステップ１５０７）。ＯＳコンテクストは、実行オペレーティングシステムを切り替えるときに参照するデータ構造で、ページテーブルアドレス値とスタックポインタの初期値とで構成される。ここでは、ページテーブルレジスタ値として第二のＯＳをマップするページテーブルのアドレスを、スタックポインタ値として第二のＯＳのカーネルスタックの初期アドレスを設定する。ＯＳ識別変数１６３０には、第一のＯＳが実行中であることを示す値が格納される。ＯＳコンテクストテーブル１６１０とＯＳ識別変数１６３０については後述する。

【０１１６】次に、第二のＯＳの初期化モジュールを実行する（ステップ１５０８）。これには、オペレーティングシステム空間の切り替えが伴う。オペレーティングシステムの切り替えについては、別のフローチャートにより説明する。

【０１１７】最後に、ステップ１５０９にて、現在の割り込みテーブルレジスタ１０４に登録されている第一のＯＳの割り込みハンドラのアドレスを、割り込み識別テーブル１６２０のハンドラ欄１６２２に複写し、割り込みテーブルレジスタ値を、マルチＯＳドライバに割り当てた割り込みテーブルのアドレスに変更する。これはプロセッサ１０１の割り込みテーブルレジスタ１０４の変更により実施する。

【０１１８】割り込みテーブルをマルチＯＳドライバ内のテーブルに変更するのは、割り込み発生時にどのオペレーティングシステムが実行していても、常にプロセッサ１０１の仮想アドレス空間に割り込みテーブルが存在している必要があるためである。割り込みテーブルに登録される割り込みハンドラも、マルチＯＳドライバ内に配置される。マルチＯＳドライバの領域は、ステップ１５０３にて、第二のＯＳの仮想空間にもマッピングして共通領域２０３とするので、いつでも参照できることになる。マルチＯＳドライバの割り込み処理については後述する。

【０１１９】図１６は、共通領域２０３のうちデータ領域１６００に格納されるデータのデータ構造を示した図である。

【０１２０】１６１０は、ＯＳコンテクストテーブルである。ＯＳコンテクストテーブル１６１０は、第一のＯＳと第二のＯＳとの間の切り替えに必要なデータを保持する。本実施形態では、第一のＯＳは第二のＯＳがアイドル状態の時のみ走行できるとする。この場合、第一のＯＳ実行中のある時点で第二のＯＳへの切り替えが起こり、第二のＯＳの実行が終了した時点で、第一のＯＳに制御を戻せばよい。

【０１２１】したがって、それぞれで保存しておかなければコンテクストは一組でよい。第一のＯＳのコンテクストについては、ＯＳ切り替えが要求された時点でのページテーブルレジスタ値１６１１と、スタックポインタ値１６１２を保存しておけば、第二のＯＳ実行終了後に、第一のＯＳに制御を復帰させることができる。

【０１２２】第一のＯＳから第二のＯＳへ制御を切り替えるときには第二のＯＳは動作していない。したがって、第二のＯＳのコンテクストは、ページテーブルアドレスもスタックポインタも固定のアドレスでよい。第二のＯＳのページテーブルレジスタ１６１３とスタックポインタ値１６１４は、第二のＯＳをロードするときに設定される(ステップ１５０７)。

【０１２３】１６２０は、割り込み識別テーブルである。割り込み識別テーブル１６２０には、外部割り込みの割り込み番号毎に、どのオペレーティングシステムが割り込み処理を処理するかを示す値１６２１と、割り込みハンドラのアドレス１６２２が記録されている。外部割り込みが発生すると、共通領域２０３内の割り込みハンドラが割り込みを捕獲し、この割り込み識別テーブル１６２０の処理ＯＳ１６２１を参照して、どのオペレーティングシステムに処理させるかを決定し、ハンドラ１６２２のアドレスに制御を渡す。割り込み処理の詳細については、後述する。

【０１２４】１６３０は、実行中のオペレーティングシステムを示す値を格納しているＯＳ識別変数である。ＯＳ識別変数１６３０は、ステップ１７０１から始まるＯＳ切り替え手順でＯＳ切り替えの度に設定される。割り込み処理では、ＯＳ識別変数１６３０を参照して割り込み処理手順が決定される。

【０１２５】１６４０は、第二のＯＳの実行中に、第一のＯＳが管理しているデバイスの割り込みが発生したかを示す遅延割り込み状態変数である。遅延割り込み状態変数１６４０は、どの割り込み番号の割り込みが発生したかを記録している。ＯＳ切り替え手順は、第二のＯＳの実行が終了したときに遅延割り込み状態変数１６４０を検査して、割り込み処理を起動するかを決定する(ステップ１７０８)。

【０１２６】１６６０は、第二のＯＳで動作するデバイスドライバが、割り込み番号あるいはＩ／Ｏアドレス範囲の資源を使用要求あるいは使用解除要求したことを示す変数である。第二のＯＳが実行中に資源が要求されたとき、第一のＯＳでこの資源を使用しないように通知する必要があるが、第二のＯＳから第一のＯＳの処理を直接は呼び出すことはできない。このため、マルチＯＳドライバが第二のＯＳからの資源の要求を受け、仮予約の状態を作り、第二のＯＳの実行が終了したときに、変数１６６０を検査することによって、マルチＯＳドライバが再度、第一のＯＳに対して資源の要求を行う。第二のＯＳが実行中は、第一のＯＳは必ず停止状態にあるため、資源がマルチＯＳドライバによって仮予約の状態でも、第一のＯＳがその資源を使用することはない。

【０１２７】図１７は、本発明の実施の形態における、オペレーティングシステムの切り替え手順を示すフローチャートである。この切り替えの手順は、第一のＯＳの実行中に呼び出され、第二のＯＳへの切り替えを行う。

【０１２８】図１７に示した手順は、第二のＯＳへの切り替え後に実行する第二のＯＳもモジュールのアドレスと、そのモジュールへ渡す引数を引数として受け取る。第二のＯＳのモジュールのアドレスは、ステップ１５０５で取得しているものとする。

【０１２９】まず、ステップ１７０１で、現在のスタックポインタ値とページテーブルレジスタ値を、ＯＳコンテクストテーブル１６１０の、第一のＯＳのコンテクストとして保存する。ステップ１７０１では、現在のスタックポインタ値を１６１２に、現在のページテーブルレジスタ値を１６１１に保存する。他のレジスタコンテクストについては、ＯＳコンテクストテーブル１６１０に保存する必要はない。必要があれば、第一のＯＳのスタックに保存すればよい。

【０１３０】スタックポインタ値とページテーブルレジスタ値を保存した後、ステップ１７０２にて、ページテーブルレジスタ１０５に第二のＯＳを仮想空間にマップするページテーブルのアドレスを設定する。これは、ＯＳコンテクストテーブル１６１０の１６１３に記録されている。更に、スタックポインタを第二のＯＳ用に設定する。これも、テーブル１６１０の第二のＯＳのスタックポインタ１６１４に格納されている。

【０１３１】次のステップ１７０３で、第一のＯＳの割り込み状態を示す遅延割り込み状態変数１６４０と割り込み資源の要求変数１６６０をクリアし、割り込み管理テーブル９００とＩ／Ｏアドレス管理リスト９１０の内容を共通領域２０３に複写しておく。そして、現在実行中のＯＳを示しているＯＳ識別変数１６３０を、第二のＯＳを示す値に書き換える(ステップ１７０４)。スタックポインタ、ページテーブルレジスタ１０５、および、ＯＳ識別変数ｉ６３０は、常に一貫した値になっていなければならないので、ここまでのステップ１７０１、ないし、１７０４は、全て外部割り込みを禁止した状態で実行しなければならない。

【０１３２】続くステップ１７０５で、引数として渡されたモジュールのアドレスに制御を移し、第二のＯＳに制御を渡す。本実施形態においては、第一のＯＳは第２のＯＳが動作していないとき、つまり、第二のＯＳの処理が終了したときに、ステップ１７０６へ制御が戻る。

【０１３３】ステップ１７０６では、ステップ１７０１でＯＳコンテクストテーブル１６１０に保存したページテーブルレジスタ値１６１１と、スタックポインタ値１６１２のそれぞれを回復する。続くステップ１７０７で、ＯＳ識別変数１６３０を第一のＯＳが実行中であることを示す値に変更する。この２つステップの処理も割り込みを禁止した状態で実行しなければならない。

【０１３４】次に、第二のＯＳの実行中に発生した、第一のＯＳが管理するデバイスの外部割り込みを処理する。まず、ステップ１７０８では、遅延割り込み状態変数１６４０を検査して、割り込みが発生したかを検査する。発生していない場合は、ステップ１７１１を実行する。割り込みが発生している場合は、ステップ１７０９を実行する。このステップでは、第二のＯＳが実行中に発生した割り込みを、第一のＯＳが管理している遅延割り込み状態変数に、未処理の割り込みがある旨を記録する。続いて、第一のＯＳの割り込み処理を起動する(ステップ１７１０)。全ての割り込み処理が終了したとき、ステップ１７１１を実行する。

【０１３５】ステップ１７１１では、第二のＯＳ実行中にデバイスドライバのロードあるいはアンロードによりハードウェアデバイスが必要とする割り込み番号あるいはＩ／Ｏアドレス範囲の資源の利用状態に変化があったかを検査する。つまり、第二のＯＳを実行中に、図１０で説明した処理手順により、第二のＯＳが使用するハードウェアデバイスを制御するため、対応するデバイスドライバのロードが行われ、資源の使用要求がデバイスドライバから第二のＯＳに通知された場合、第二のＯＳはマルチＯＳドライバに対して資源の使用が要求されたことを通知する。通知を受けたマルチＯＳドライバは、ステップ１７０３で共通領域２０３に複写した割り込み管理テーブル９００とＩ／Ｏアドレス管理リスト９１０を調査し、資源の使用が可能ならば、共通領域２０３のテーブル（９００，９１０）を更新後、第二のＯＳに対して資源の使用の許可を出す。同時に、資源の使用の状態が変化したことを、変数１６６０に記録しておく。第二のＯＳを実行中にデバイスドライバのアンロードが行われた場合と同様であり、当該デバイスドライバから第二のＯＳへ、更に、第二のＯＳからマルチＯＳドライバへと要求が伝わり、変数１６６０に変化を記録する。

【０１３６】ステップ１７１１では、変数１６６０を検査することによって、割り込み番号あるいはＩ／Ｏアドレス範囲の資源の利用状態に変化があったことを検出し、変化があった場合は、共通領域２０３にある割り込み管理テーブル９００とＩ／Ｏアドレス管理リスト９１０の内容を、第一のＯＳが管理する割り込み管理テーブル９００とＩ／Ｏアドレス管理リスト９１０へと書き戻す(ステップ１７１２)。

【０１３７】これにより第二のＯＳが使用するハードウェアデバイスの資源は、第一のＯＳにも設定されるため、第一のＯＳのデバイスドライバにこの資源が割り当てられることが防げる。第一のＯＳから見て、第二のＯＳが使用するハードウェアデバイスの資源は、第一のＯＳにロードされているマルチＯＳドライバが使用する資源として見せることによって、オペレーティングシステム内の管理の一貫性は保たれる。

【０１３８】図１８は、本実施形態の割り込み処理手順を示すフローチャートである。この手順を実行するモジュールは、割り込みハンドラとしてプロセッサ１０１の割り込みテーブル１０７に登録される。さらに、この割り込みハンドラは、どのオペレーティングシステムからも参照できる共通領域２０３に配置される。

【０１３９】外部割り込みが発生して、プロセッサ１０１により割り込みハンドラが起動されると、割り込みハンドラは割り込み要因を検査し、割り込みを発生したハードウェアデバイスが第一のＯＳが関するハードウェアデバイスか、第二のＯＳが管理するハードウェアデバイスかを判定する(ステップ１８０１)。この判定は、割り込み識別テーブル１６２０を割り込み番号をインデックスとしてＯＳ欄１６２１を参照することにより実施される。第一のＯＳのハードウェアデバイスである場合はステップ１８０２へ、第二のＯＳのハードウェアデバイスである場合はステップ１８０５へ進む。例えば、図１６でいえば、割り込み番号が１であれば第一のＯＳの割り込みであり、割り込み番号４であれば第二のＯＳの割り

込みとなる。

【０１４０】割り込みが第一のＯＳのハードウェアデバイスの割り込みである場合、ステップ１８０２を実行する。ステップ１８０２では、割り込み発生時に実行していたオペレーティングシステムを判定する。この判定は、ＯＳ識別変数１６３０を参照して実施する。実行中のＯＳが第一のＯＳの場合はステップ１８０３へ、第二のＯＳの場合はステップ１８０４へ進む。

【０１４１】ステップ１８０３から始まる処理は、第一のＯＳが管理しているハードウェアデバイスが、第一のＯＳを実行中に割り込みを発生した場合の処理である。

【０１４２】ステップ１８０３では、あたかも１８０１から始まる処理が存在せず、第一のＯＳの割り込みハンドラが、直接プロセッサ１０１から制御を受けたように見えるようにコンテクストを設定する。ここでコンテクストとは、スタックの内容やレジスタの内容を示す。そして、第一のＯＳの割り込みハンドラのアドレスは、割り込み識別テーブル１６２０のハンドラ欄１６２２に格納されている。例えば、割り込み番号１の割り込みならば、１をインデックスとして割り込み識別テーブルを参照して、ハンドラのアドレスを求める。この場合、ステップ１８０１から始まる手順には制御は戻らず、第一のＯＳが処理を続ける。

【０１４３】第一のＯＳが管理しているハードウェアデバイスが、第二のＯＳを実行中に割り込みを発生した場合、ステップ１８０４を実行する。ステップ１８０４では、遅延割り込み状態変数１６４０に割り込みを発生したハードウェアデバイスの割り込み番号を記録する。割り込みハンドラの処理はこれで終了する。この場合の割り込みの処理は、実行ＯＳが第一のＯＳに切り替わったときに実行される(ステップ１７０８)。

【０１４４】発生した外部割り込みが、第二のＯＳが管理するハードウェアデバイスの割り込みだった場合、ステップ１８０５へ進み、どちらのＯＳが実行中であるかを検査する。ここでも、ＯＳ識別変数１６３０によって実行中のＯＳを判定する。第一のＯＳが実行中の場合は、ステップ１８０６へ、第二のＯＳが実行中の場合はステップ１８１１へ進む。

【０１４５】第二のＯＳが管理するハードウェアデバイスの割り込みが、第二のＯＳの実行中に発生した場合、ステップ１８１１を実行する。ステップ１８１１は、第二のＯＳの割り込みハンドラを起動する。第二のＯＳの割り込みハンドラのアドレスは、割り込み識別テーブル１６２０のハンドラ欄１６２２に記録されている。第二のＯＳの割り込みハンドラ処理が終了して戻ってきたら、この割り込みハンドラも終了し、割り込まれたときのコンテクストを回復して制御を戻す。

【０１４６】第二のＯＳが管理するハードウェアデバイスの外部割り込みが、第一のＯＳの実行中に発生した場合、ステップ１８０６を実行する。この場合は、第一のＯＳの実行よりも第二のＯＳの処理を優先して実行する。

【０１４７】まず、ステップ１８０６では、コンテクストを保存する。ここでのコンテクストとは、割り込み処理が終了した後で第一のＯＳに制御を戻すときに、割り込まれたときの状態を回復するのに必要なスタックの内容とレジスタの内容を示す。このコンテクストは、第一のＯＳのカーネルのスタックに保存される。

【０１４８】続いて、実行ＯＳの切り替えと第二のＯＳの割り込み処理の起動を実行する(ステップ１８０７，１８０８)。これは、ステップ１７０１から始まる手順により実行する。

【０１４９】第二のＯＳの処理が終了した時点で、第一のＯＳへの切り替えを実行し(ステップ１８０９)、割り込み時のコンテクストを回復し(ステップ１８１０)、第一のＯＳの処理を再開する。ステップ１８０９の処理は、必ずしも、ステップ１８０１から始まる処理と同一のもモジュールで実行されなくてもよい。第一のＯＳへの切り替えにより処理はこのモジュールに復帰する。

【０１５０】２つのオペレーティングシステムで共有しているクロック割り込みの処理について説明する。

【０１５１】クロック割り込みは、共通領域２０３内の割り込みハンドラにより捕獲する。この割り込みのハンドラでは、まず、第二のＯＳのクロック割り込み用の割り込みハンドラを実行する。第二のＯＳの割り込みハンドラはハンドラ欄１６２３に格納されている。第二のＯＳの割り込みハンドラが終了したら、図１８のステップ１８０２から始まる処理により第一のＯＳの割り込み処理を実行する。第１のＯＳの割り込みハンドラのアドレスはハンドラ欄１６２２に格納されている。

【０１５２】以上により、一台の計算機で２つのオペレーティングシステムを同時に動作させることが可能になる。

【０１５３】さらに、上述した実施形態において、図１５で示した第二のＯＳのロード手順に図１９のフローチャートで示すようなステップ１５１０を加えることによって、第二のＯＳに第一のＯＳの動作状態を監視させることが可能になる。図１９の処理手順について以下説明する。

【０１５４】図１９の処理手順のうち、ステップ１５０１からステップ１５０９までは、図１５のフローチャートの処理と同じであり、新たに監視処理を実施するためのステップ１５１０を追加している。

【０１５５】オペレーティングシステムは、実行中に障害が発生し、処理の継続が不可能と判定した場合、オペレーティングシステムのエラー処理関数を実行することによって、計算機に接続している全てのハードウェアデバイスを停止させ、最後に計算機のプロセッサを停止させる。本実施形態では、一台の計算機上で複数のオペレーティングシステムを動作させているため、一つのオペ

レーティングシステムの障害で、必ずしも計算機全体を停止させる必要はなく、障害を起こしたオペレーティングシステム以外のオペレーティングシステムが計算機の動作を継続することが可能である。

【０１５６】本実施形態では、第一のＯＳに障害が発生し、第二のＯＳがこれを検出するものとする。ステップ１５１０では、第二のＯＳのローディング処理の最後に、第一のＯＳが呼び出すエラー処理関数の先頭の命令を、割り込みを発生する命令に書き換える。例えば、図１６に示した割り込み識別テーブル１６２０の状態で、割り込み番号５を発生させる命令を書き込むと、第一のＯＳで継続不能な障害が発生した場合、第一のＯＳのエラー処理関数が実行される前に割り込み番号５がプロセッサ１０１で発生する。割り込み番号５は、第二のＯＳが管理する割り込みであり、割り込み番号５のハンドラ欄１６２３の内容にしたがって、第２のＯＳのハンドラが呼び出される。つまり、第一のＯＳのエラー処理関数が実行される前に、第二のＯＳは第一のＯＳに障害が発生したことを検知できる。割り込み番号５の割り込みハンドラ処理で、マルチＯＳドライバに対して、第一のＯＳがダウンしたことを通知し、以後、第二のＯＳのみを動作させるようにすれば、第二のＯＳの処理は継続させることができる。処理を継続する第二のＯＳが、第一のＯＳの障害診断や障害回復を行い、オペレータに対する通知を行うことが可能である。これにより、一台の計算機で複数のオペレーティングシステムを同時に動作させる環境において、１つのオペレーティングシステムが他のオペレーティングシステムで発生した障害を監視するシステムを実現できる。

【０１５７】また、上述した実施形態では、一台の計算機で複数のオペレーティングシステムを動作させる機能を、マルチＯＳドライバというデバイスドライバで実施したが、計算機のブート処理によりロードされるオペレーティングシステムの内部にマルチＯＳを実施する機能を組み込むことも可能である。この実施形態では、図６のブート処理のステップ６１１の処理の中で、図１３に示したマルチＯＳを実施する処理(ステップ１３０１から１３０６)を実行することによって、第一のＯＳが立ち上がった時点で、第二のＯＳのローディングが可能な状態になる。この実施形態においても、図１５、図１６、図１７、図１８、図１９に示した処理手順は変更することはない。これにより、１台の計算機で、自オペレーティングシステム以外のオペレーティングシステムを同時に動作させることができるオペレーティングシステムを実現し、複数のオペレーティングシステムを同時に動作させることができる。

【０１５８】

【発明の効果】以上説明した実施形態によれば、特別なハードウェアの支援なしで、１台の計算機で複数のオペレーティングシステムを同時に動作できる。第一のＯＳに欠けている機能を容易に導入することができ、さらに、その機能を第一のＯＳとはまったく独立して動作させることができる。

【０１５９】また、マルチＯＳを実施する機能をデバイスドライバとして実現することにより、第一のＯＳの処理を変更することがないため、既存のオペレーティングシステムに対して容易に機能を追加できる。さらに、１台の計算機で複数のオペレーティングシステムを同時に動作させた環境で、１つのオペレーティングシステムは他のオペレーティングシステムの障害を監視することができ、監視オペレーティングシステムが被監視オペレーティングシステムの障害診断や障害回復を行うことによって、計算機全体の信頼性および保守性を向上されることができる。

【０１６０】さらにまた、ハードウェアデバイスが使用する割り込み資源を、複数のオペレーティングシステム間で動的に割り当てることによって、１つのオペレーティングシステムに割り当てたハードウェアデバイスを、該オペレーティングシステムが利用終了後、他のオペレーティングシステムが利用することが可能になる。

【０１６１】

【発明の効果】本発明によれば、特別なハードウェアを用いる必要なく、また、入出力処理等の特権命令のエミュレーションを行うことなく１台の計算機上で複数のオペレーティングシステムを同時、並列的に動作させることができる。

【図面の簡単な説明】

【図１】本発明の一実施形態における計算機構成を示すブロック図である。

【図２】計算機構成を示す模式図である。

【図３】ページテーブルの構成を示す図である。

【図４】割り込みテーブルの構成を示す図である。

【図５】割り込み制御装置の構成を示す図である。

【図６】計算機のブート処理手順を示すフローチャートである。

【図７】第一のＯＳのカーネル構成情報ファイルの構成を示す図である。

【図８】カーネル起動パラメータのデータ構造を示す図である。

【図９】第一のＯＳのデバイス管理テーブルのデータ構造を示す図である。

【図１０】オペレーティングシステムのデバイスドライバのロード処理手順を示すフローチャートである。

【図１１】オペレーティングシステムのデバイスドライバのアンロード処理手順を示すフローチャートである。

【図１２】マルチＯＳドライバの処理構造を示す図である。

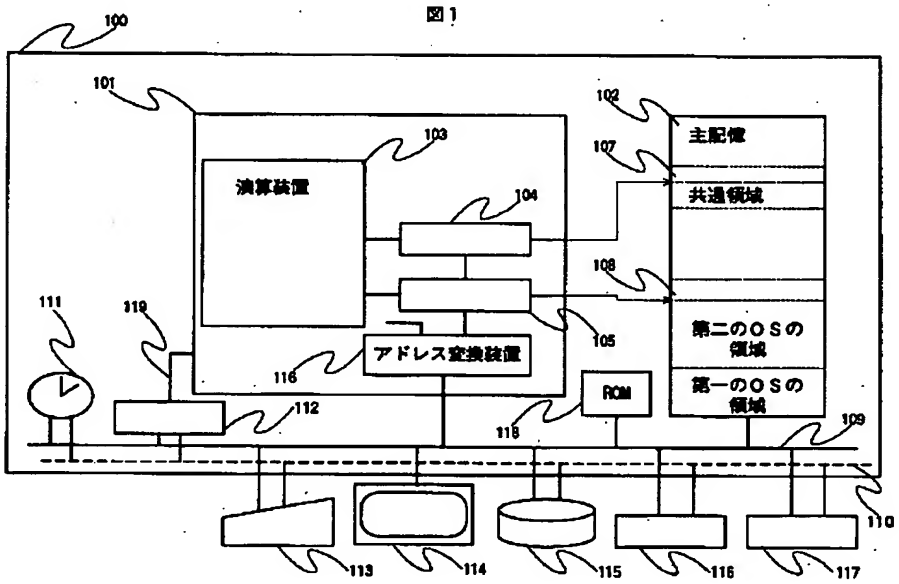
【図１３】マルチＯＳドライバのロード処理手順のフローチャートである。

【図１４】仮想アドレス空間と主記憶装置１０２の物理

メモリの対応を示す図である。
【図１５】第二のＯＳのロード処理手順を示すフローチャートである。
【図１６】第一のＯＳと第二のＯＳが共有するデータ構造を示す図である。
【図１７】実行ＯＳの切り替え処理手順を示すフローチャートである。
【図１８】割り込み処理手順を示すフローチャートである。
【図１９】第２のＯＳに監視機能を付加するための、第二のＯＳのロード処理手順を示すフローチャートである。
【符号の説明】
１００･･･計算機
１０１･･･プロセッサ
１０２･･･主記憶装置
１０３･･･演算装置
１０４･･･割り込みテーブルレジスタ
１０５･･･ページテーブルレジスタ
１０６･･･アドレス変換装置
１０７･･･割り込みテーブル
１０８･･･ページテーブル
１０９･･･バス
１１０･･･信号線
１１１･･･クロック割り込み生成器
１１２･･･割り込みコントローラ
１１３･･･キーボード
１１４･･･ディスプレイ
１１５･･･磁気ディスク
１１６、１１７･･･外部機器
１１８･･･記憶装置
１１９･･･割り込みバス

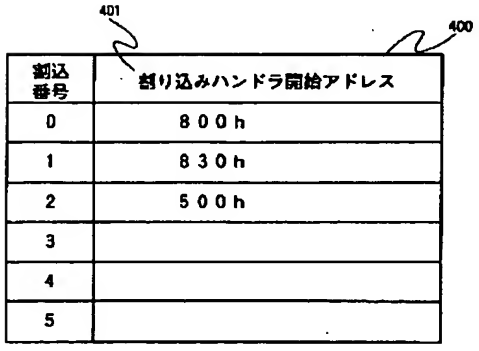
【図１】

図１



【図４】

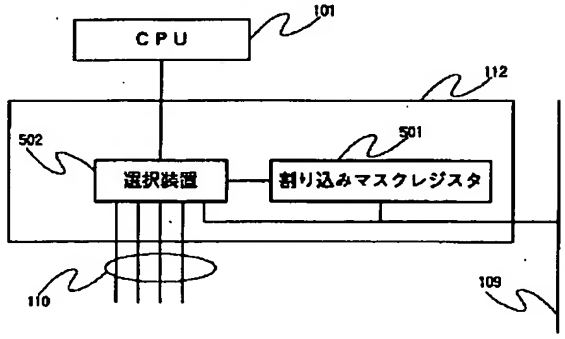
図４



割込番号	割り込みハンドラ開始アドレス
0	８００ｈ
1	８３０ｈ
2	５００ｈ
3	
4	
5	

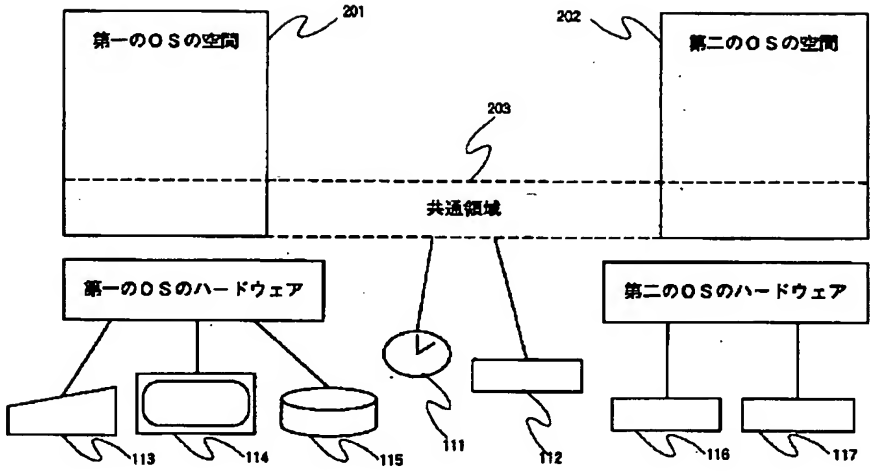
【図５】

図５



【図２】

図２



【図３】

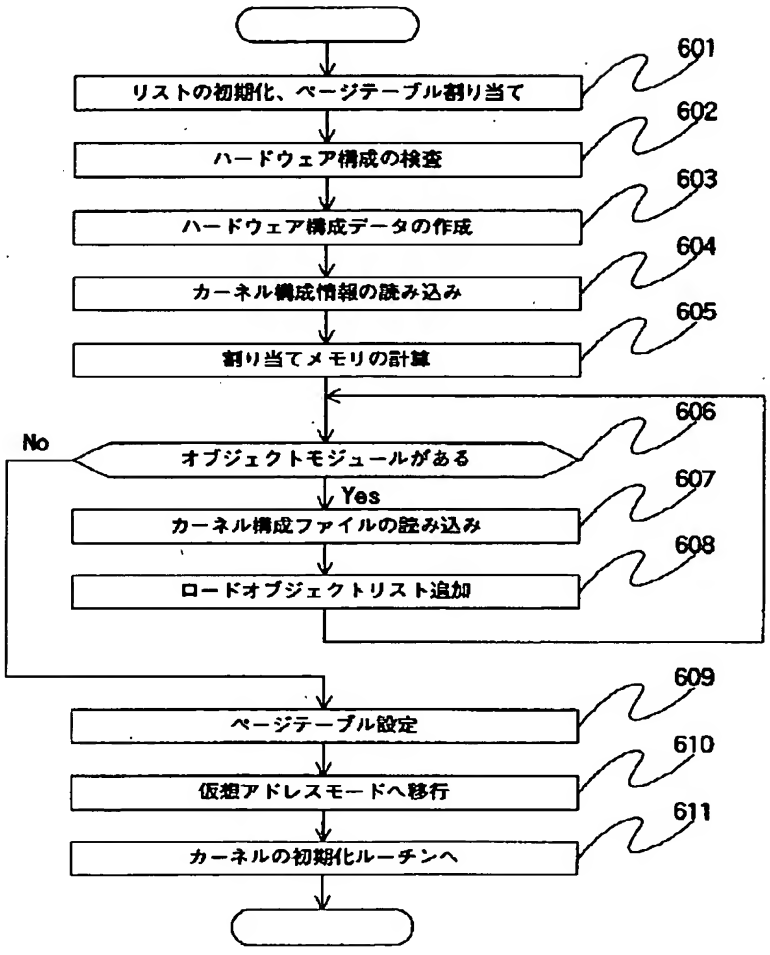
図３

仮想ページ番号	有効	物理ページ番号
0	✓	0
1	✓	5 6
2	✓	2 3
3		
4		
5	✓	3 2
		⋮

301　302　300

【図６】

図６



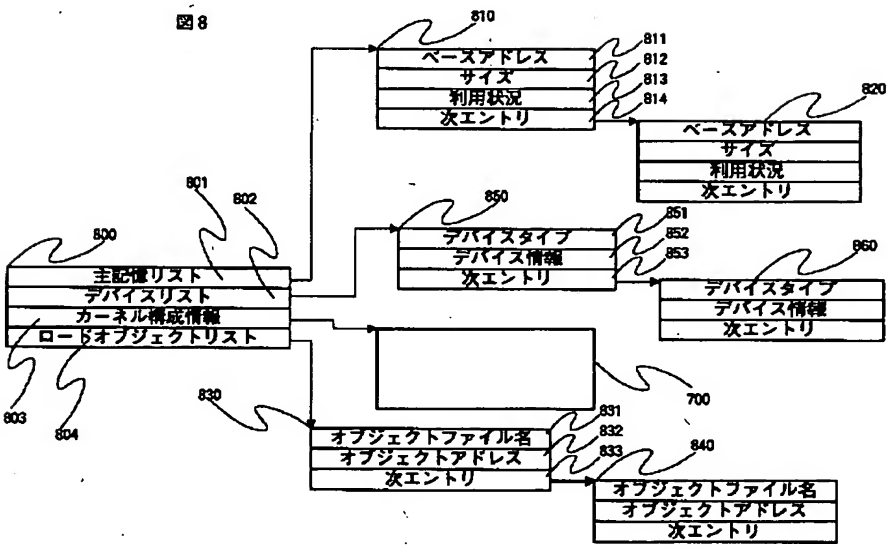
【図７】

図７

名　前	内　容
メモリサイズ	数値
オブジェクトファイル	kernel, driver1, driver2
driver2	driver2 固有データ
Secondary OS	第二のＯＳの固有データ

700, 701, 703, 705

【図８】

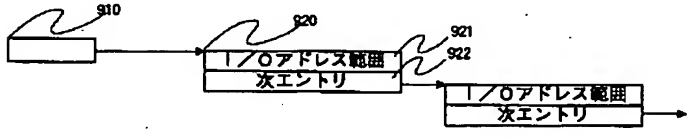


【図９】

図９

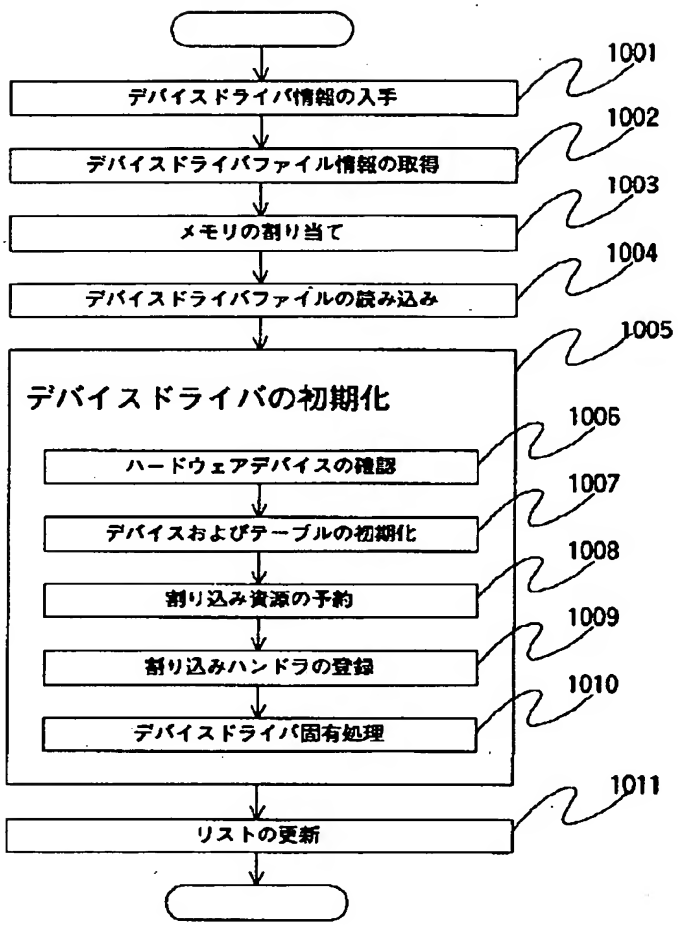
割り込み番号	使用
0	
1	
2	
3	
4	
5	

900, 901, 902



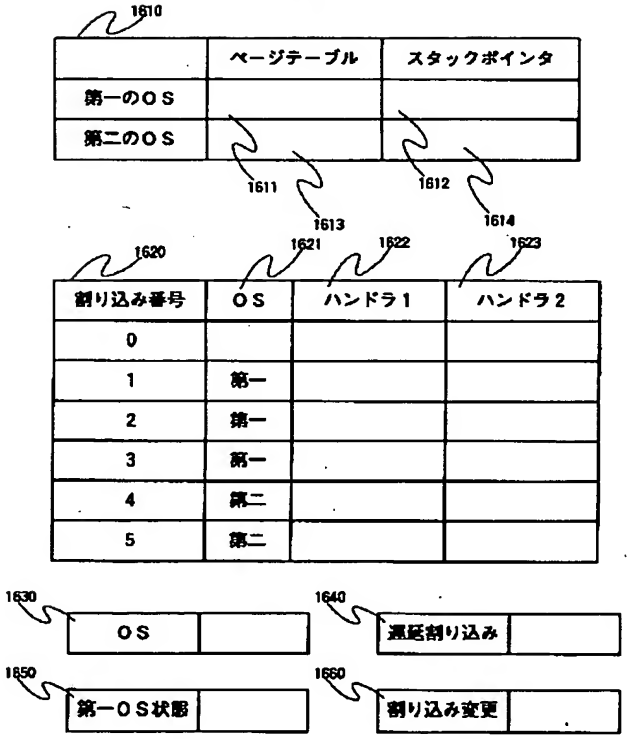
【図１０】

図１０



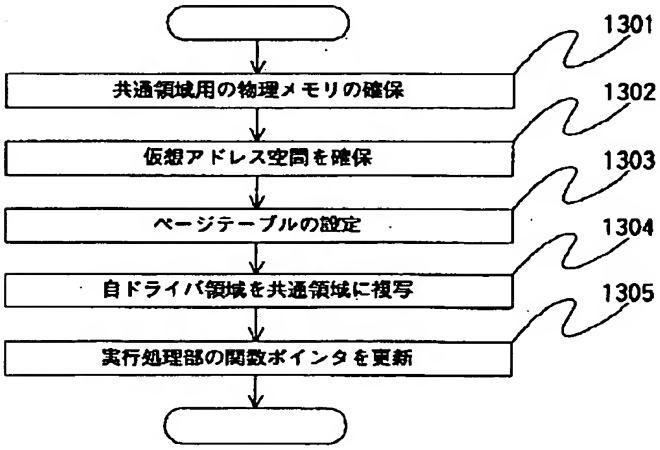
【図１６】

図１６



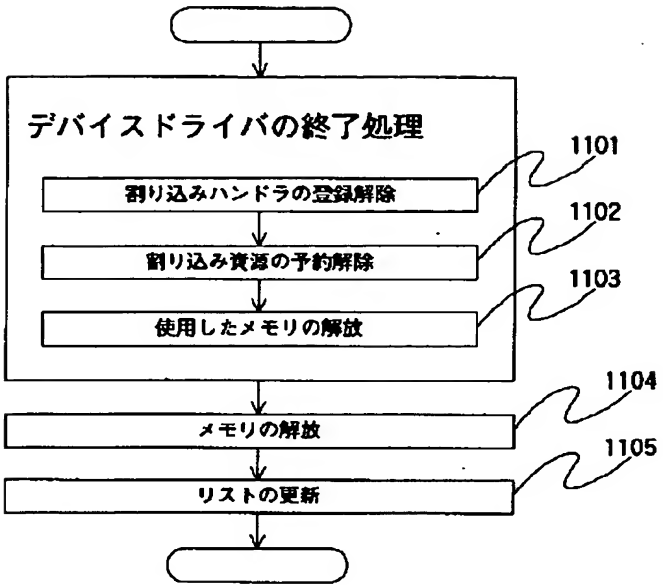
【図１３】

図１３



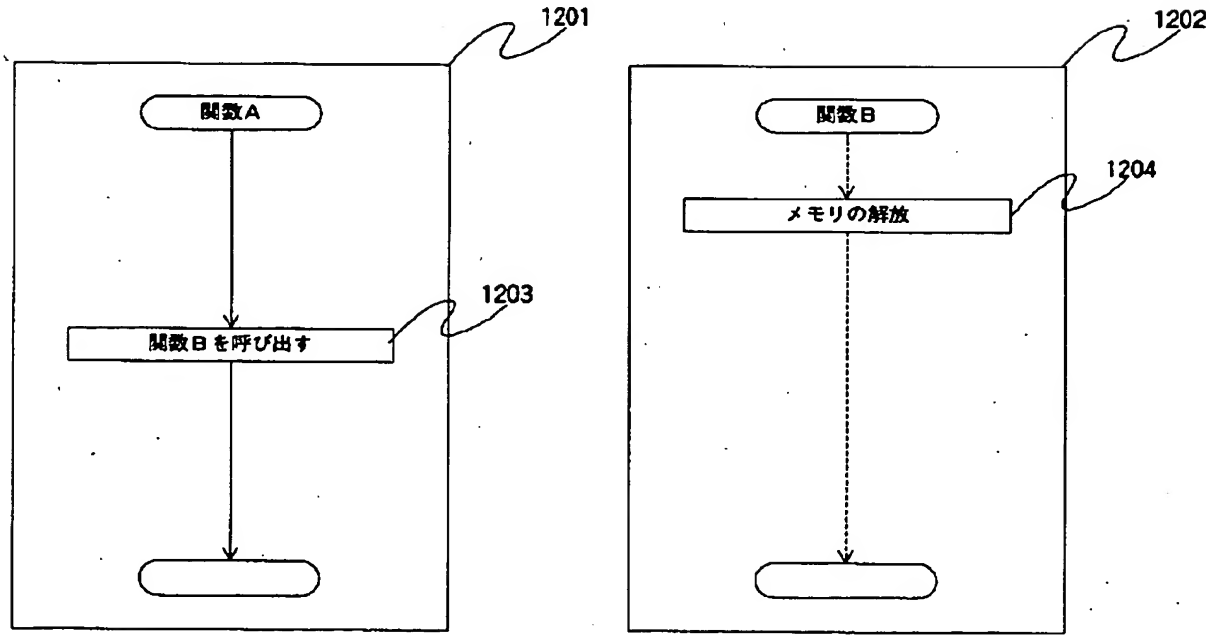
【図１１】

図１１



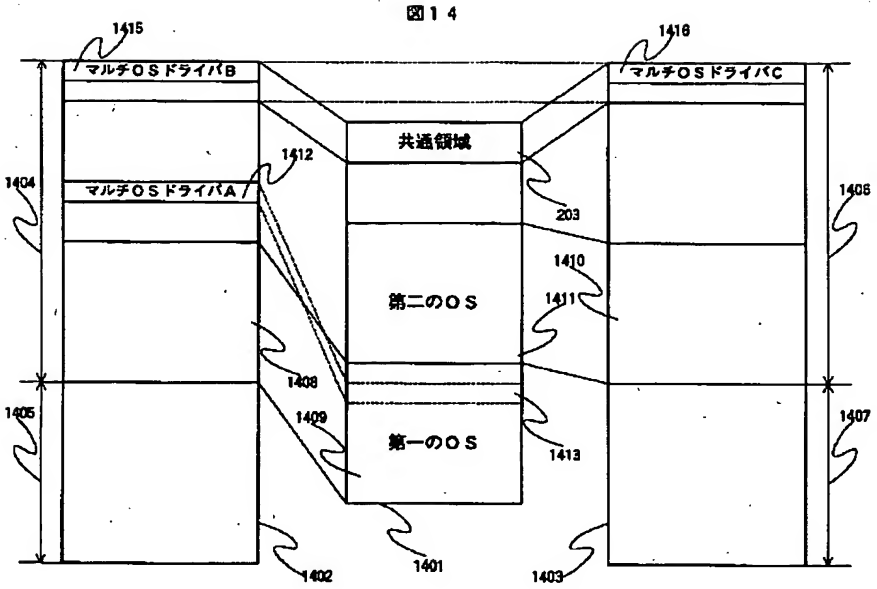
【図１２】

図１２



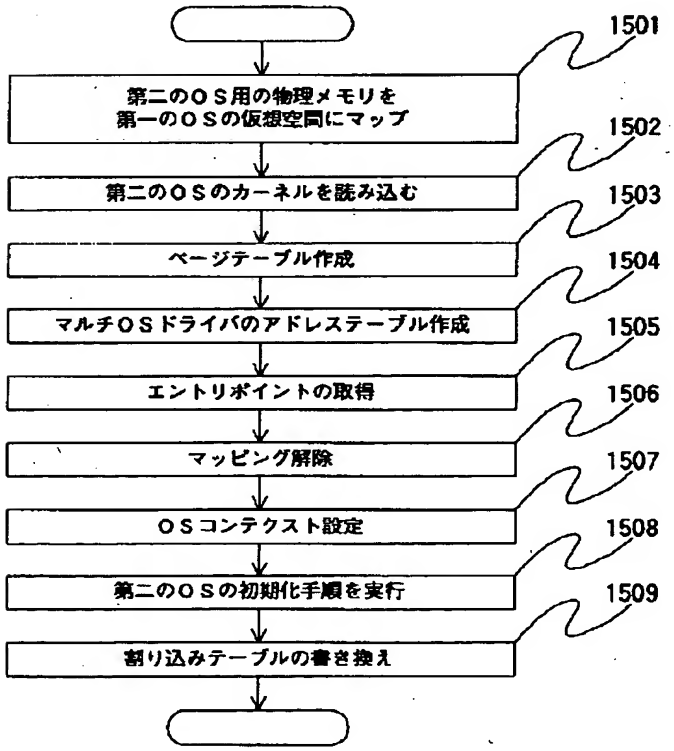
【図１４】

図１４



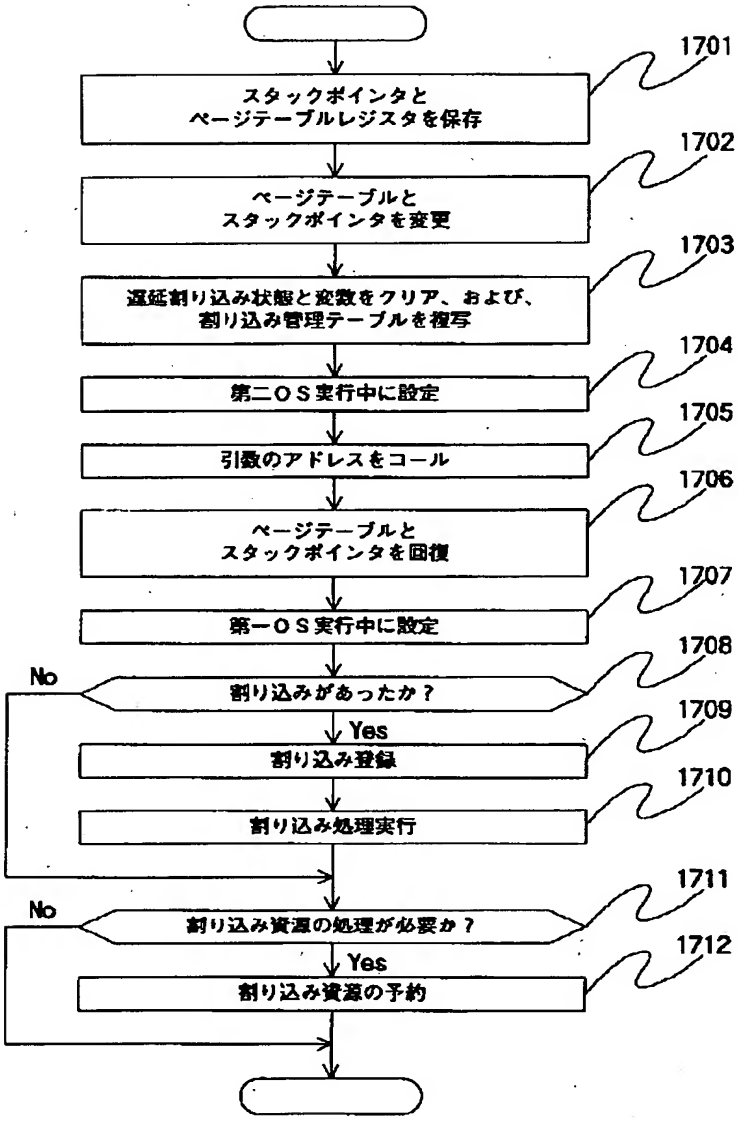
【図１５】

図１５



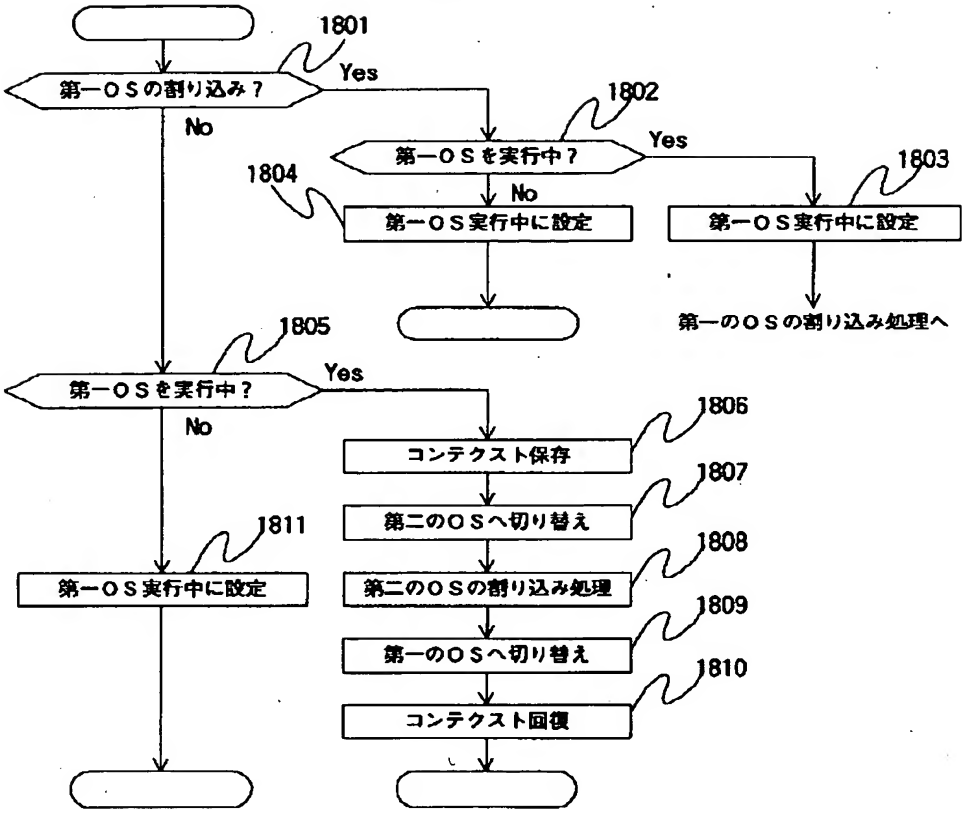
【図１７】

図１７


1701
スタックポインタと
ページテーブルレジスタを保存
1702
ページテーブルと
スタックポインタを変更
1703
遅延割り込み状態と変数をクリア、および、
割り込み管理テーブルを複写
1704
第二ＯＳ実行中に設定
1705
引数のアドレスをコール
1706
ページテーブルと
スタックポインタを回復
1707
第一ＯＳ実行中に設定
1708
No
割り込みがあったか？
Yes
1709
割り込み登録
1710
割り込み処理実行
1711
No
割り込み資源の処理が必要か？
Yes
1712
割り込み資源の予約

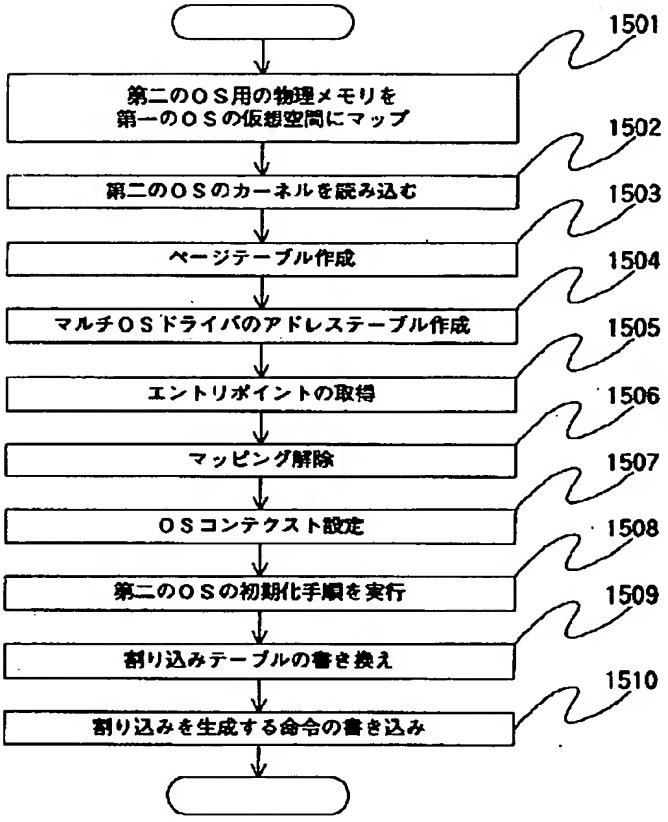
【図１８】

図１８


1801
第一ＯＳの割り込み？
Yes
No
1802
第一ＯＳを実行中？
Yes
No
1803
1804
第一ＯＳ実行中に設定
第一ＯＳ実行中に設定
第一のＯＳの割り込み処理へ
1805
第一ＯＳを実行中？
Yes
No
1806
コンテクスト保存
1807
第二のＯＳへ切り替え
1808
第二のＯＳの割り込み処理
1809
第一のＯＳへ切り替え
1810
コンテクスト回復
1811
第一ＯＳ実行中に設定

【図１９】

図１９



フロントページの続き

(72)発明者　佐藤　雅英
神奈川県川崎市麻生区王禅寺1099番地　株式会社日立製作所システム開発研究所内
(72)発明者　梅都　利和
愛知県尾張旭市晴丘町池上１番地　株式会社日立製作所情報機器事業部内

Ｆターム(参考)　5B042 GA22 GA23 GA25 GC07 GC12 JJ01 JJ05
5B076 AA13 AA14 AB14 CA08
5B098 BA06 BB11 EE02 EE06 GA02 GC01 GD02 GD20 GD22 HH04 HH07 JJ08